

Managing Your Dotfiles With Git

Simplifying the backup process



Eric Chi

Follow

Aug 17 · 7 min read ★



DOTFILES

Image by the author

Complete source code can be found here:

<https://github.com/ericjaychi/sample-dotfiles>

In today's article, we will be talking about backing up your lovely dotfiles. These include files like `.vimrc`, `.bashrc`, and `.gitconfig`. Keep in mind, this is simply how I back them up. Feel free to back them up however you want. After all, I am just some random stranger on the internet.

This will allow you to easily pull your dotfiles onto any machine, as long it's a UNIX-based system, as well as keeping your dotfiles always backed up.

For reference, I used this blog post from Michael Smalley's blog and made a few adjustments for my misc files I want to backup.

The original source can be found here:

<http://blog.smalleycreative.com/tutorials/using-git-and-github-to-manage-your-dotfiles/>. I highly recommend you check it out, as I have been using this way of backing up my dotfiles for quite some time now and use this almost daily. It also saves me a ton of time when I get a new computer.

Table of Contents

- Step 1. Create a repository
- Step 2. Add your dotfiles into the repository
- Step 3. The make-symlinks.sh script
- Step 4. Update your dotfiles

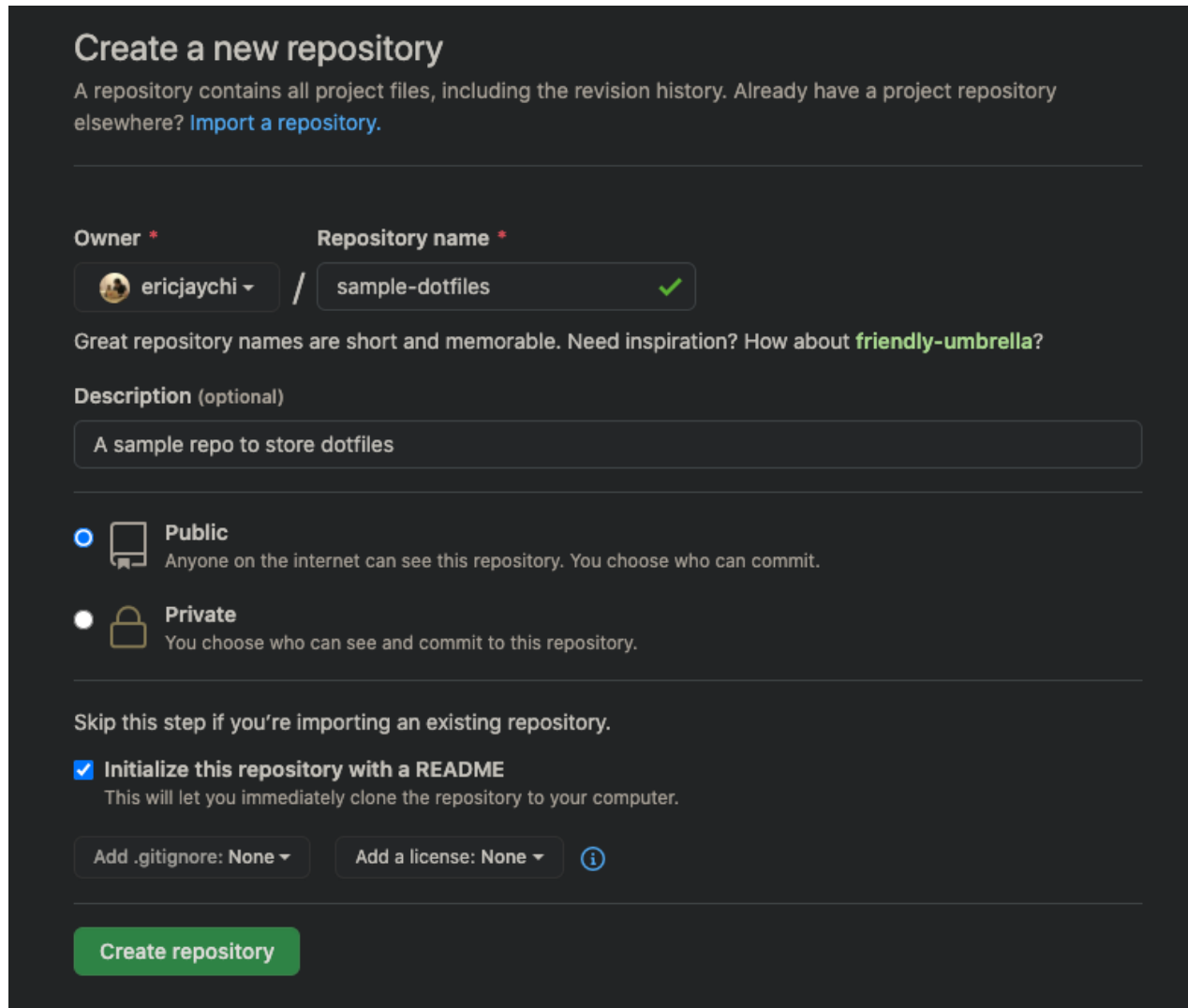
Step 1. Create a Repository

I like to make a lot of assumptions, which is bad, but this case is no different. I'm going to assume you know what Git is. If you don't know what Git is, I'd suggest you look into it as it's a very important tool for programmers.

To begin, I'm going to create a new repository called `sample-dotfiles`. Keep in mind, you can name this repo whatever you like.

I will be using GitHub in this article, but feel free to use whatever service you like for managing repositories. Keep in mind that if you have sensitive information in your dotfiles, then I would suggest making this repo private. Or if you just don't want anyone

else to see it, I would suggest making the repo private.

The image shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main input fields: 'Owner' and 'Repository name'. The 'Owner' field is set to 'ericjaychi' and the 'Repository name' field is set to 'sample-dotfiles'. A green checkmark is visible next to the repository name. Below these fields, there is a 'Description (optional)' field with the text 'A sample repo to store dotfiles'. Underneath the description, there are two radio button options for repository visibility: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is described as 'You choose who can see and commit to this repository.' Below the visibility options, there is a section for 'Skip this step if you're importing an existing repository.' followed by a checked checkbox for 'Initialize this repository with a README', which is described as 'This will let you immediately clone the repository to your computer.' At the bottom, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None', followed by an information icon. A large green 'Create repository' button is at the very bottom.

Creating a new repo in GitHub

Alright, once we have created our new repo for our dotfiles, let's go ahead and clone it onto our machine by copying the clone button on GitHub.

```
git clone git@github.com:{github_user_name}/sample-dotfiles.git
```

Or if you are using HTTPS:

```
git clone https://github.com/{github_user_name}/sample-dotfiles.git
```

Now that you have your repo all set up, you'd ask yourself, "Oh, don't we just add everything into this repo and send it up?" Yes and no. We are going to want to keep our dotfiles in their original place, which is usually in our home directory of a user. We are going to use this repo to store symlinks to our original dotfiles, which will make it seem like nothing has changed. It's programming magic at its finest.

Step 2. Add Your Dotfiles Into the Repository

Depending on how you have your dotfiles set up, they could be all in one file or they could be split up into different files, like mine. Regardless of how you have your dotfiles set up, just grab all of those you wish to back up. Typically they're in a directory like `/Users/eric-chi` if you're on a Mac or `/home/eric-chi` if you're on

a machine with Linux installed.

In this example, I wish to back up a simple `.bashrc`, `.functions`, and a `.gitconfig`. If you have other items, like a `.vimrc` or `.aliases`, feel free to add them in the repo. Remember, this process is the same for every dotfile you want to add.

I also decided to remove the `.` in front of the files for simplicity and for me to tell the difference between the actual dotfiles and the ones inside of the repo.

Remember, we are copying these into the repo. We are not moving them!

```
KOHRUYU 22:18 @ -/Dropbox/___igor/repos/blogs/sample-dotfiles (master) $ ll
total 40
drwxr-xr-x@ 7 eric-chi  staff   224 Aug  9 22:18 ./
drwxr-xr-x@ 5 eric-chi  staff   160 Aug  9 22:15 ../
drwxr-xr-x@ 12 eric-chi  staff   384 Aug  9 22:15 .git/
-rw-r--r--@ 1 eric-chi  staff    50 Aug  9 22:15 README.md
-rw-r--r--@ 1 eric-chi  staff  4288 Aug  9 22:18 bashrc
-rw-r--r--@ 1 eric-chi  staff  1045 Aug  9 22:18 functions
-rw-r--r--@ 1 eric-chi  staff  1928 Aug  9 22:18 gitconfig
KOHRUYU 22:18 @ -/Dropbox/___igor/repos/blogs/sample-dotfiles (master) $ |
```

So this is what I have inside of my repo right now. It's pretty simple so far — we aren't doing anything too crazy. The beauty of this approach is that we will be using symlinks. Our computer won't recognize the difference when we load these files. It's going to treat them as if they were on the user root directory, even though we are going to tuck our files inside of our repo. Now we

are going to use Michael Smalley's script to populate these symlinks.

Step 3. The make-symlinks.sh Script

This is really where the magic comes in. Like I mentioned before, we are simply creating symlinks on our home directory to sync up to our dotfiles repo. Nothing else. To do that, we will be using this script:

```
1  #!/bin/bash
2  #####
3  # .make.sh
4  # This script creates symlinks from the home directory to any desired dotfil
5  #####
6
7  ##### Variables
8
9  # dotfiles directory
10 dir=~/.Users/eric-chi/Dropbox/___igor/repos/blogs/sample-dotfiles
11
12 # old dotfiles backup directory
13 olddir=~/.Users/eric-chi/Dropbox/___igor/repos/blogs/sample-dotfiles/dotfiles_
14
15 # list of files/folders to symlink in homedir
16 files="bashrc gitconfig functions"
17
18 #####
19
20 # create dotfiles_old in homedir
21 echo -n "Creating $olddir for backup of any existing dotfiles in ~ ..."
```

```
22  mkdir -p $olddir
23  echo "done"
24
25  # change to the dotfiles directory
26  echo -n "Changing to the $dir directory ..."
27  cd $dir
28  echo "done"
29
30  # move any existing dotfiles in homedir to dotfiles_old directory, then create
31  for file in $files; do
32      echo "Moving any existing dotfiles from ~ to $olddir"
33      mv ~/.$file ~/dotfiles_old/
```

Create a file, `make-symlinks.sh` and tuck this code inside of the dotfiles repo.

To explain what this is doing, this script has a few variables defined that allow us to customize the targeted files as well as the directories that will be used during this script.

- `dir` = The path to your dotfiles repository that we have created
- `olddir` = A copy of your old dotfiles after the script runs. It'll keep a record of the old files, just in case anything happens, so that you can always revert back.
- `files` = The specific dotfiles which you want to back up

Make sure that this script is inside of the dotfiles repo so that when you push it up, it'll go with it, and then you can clone this

repo on any machine and have the script ready.

Now that we have the `make-symlinks.sh` file created, we are going to go ahead and run the script to create the symlinks.

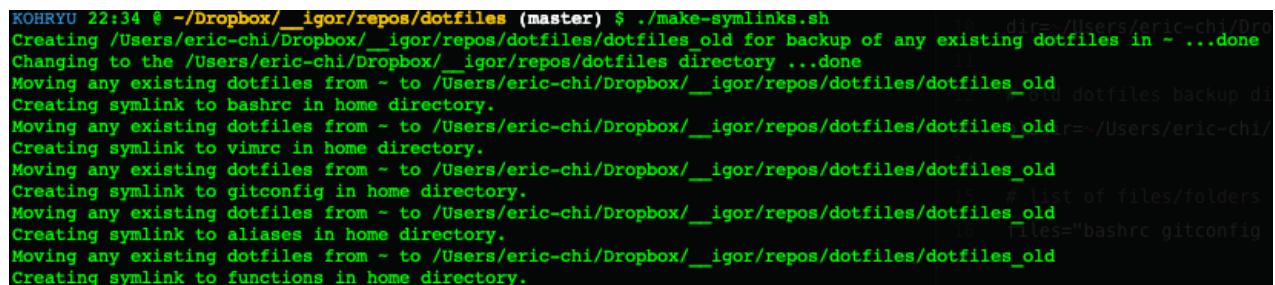
Depending on how you created the file, you may need to give it permissions so that you can execute it:

```
cd /Users/eric-chi/Dropbox/__igor/repos/blogs/sample-dotfiles
chmod +x make-symlinks.sh
```

Obviously, you're doing to want to change the path to match your path. Once you have done that, go ahead and run the script.

```
./make-symlinks.sh
```

Your terminal will run and create the respective symlinks.



```
KOHRU 22:34 @ ~/Dropbox/__igor/repos/dotfiles (master) $ ./make-symlinks.sh
Creating /Users/eric-chi/Dropbox/__igor/repos/dotfiles/dotfiles_old for backup of any existing dotfiles in ~ ...done
Changing to the /Users/eric-chi/Dropbox/__igor/repos/dotfiles directory ...done
Moving any existing dotfiles from ~ to /Users/eric-chi/Dropbox/__igor/repos/dotfiles/dotfiles_old
Creating symlink to bashrc in home directory.
Moving any existing dotfiles from ~ to /Users/eric-chi/Dropbox/__igor/repos/dotfiles/dotfiles_old
Creating symlink to vimrc in home directory.
Moving any existing dotfiles from ~ to /Users/eric-chi/Dropbox/__igor/repos/dotfiles/dotfiles_old
Creating symlink to gitconfig in home directory.
Moving any existing dotfiles from ~ to /Users/eric-chi/Dropbox/__igor/repos/dotfiles/dotfiles_old
Creating symlink to aliases in home directory.
Moving any existing dotfiles from ~ to /Users/eric-chi/Dropbox/__igor/repos/dotfiles/dotfiles_old
Creating symlink to functions in home directory.
```

I'm using my actual dotfiles repo here, but the same concept applies.

If you take a look at your home directory, whether its `/Users/eric-chi` or `/home/eric-chi`, you should see all the dotfiles we stored inside the repo have respective symlinks.

```

KOHRYU 22:40 ~ - () $ ll
total 112
drwxr-xr-x+ 37 eric-chi  staff   1184 Aug  9 22:35 ./
drwxr-xr-x   5 root     admin   160 May 27 16:30 ../
-r-----   1 eric-chi  staff    7 Aug  6 20:23 .CFUserTextEncoding
-rw-r--r--   1 eric-chi  staff 10244 Aug  9 22:19 .DS_Store
drwx-----  2 eric-chi  staff    64 Aug  9 11:17 .Trash/
lrwxr-xr-x   1 eric-chi  staff    53 Aug  9 22:34 .aliases@ -> /Users/eric-chi/Dropbox/_igor/repos/dotfiles/aliases
drwxr-xr-x   2 eric-chi  staff    64 Aug  7 21:19 .android/
drwxrwxrwx  12 eric-chi  staff   384 Aug  7 00:33 .anydesk/
-rw-----   1 eric-chi  staff  7534 Aug  9 22:39 .bash_history
-rw-r--r--   1 eric-chi  staff    42 Aug  9 11:27 .bash_profile
drwx-----  9 eric-chi  staff   288 Aug  9 11:07 .bash_sessions/
lrwxr-xr-x   1 eric-chi  staff    52 Aug  9 22:34 .bashrc@ -> /Users/eric-chi/Dropbox/_igor/repos/dotfiles/bashrc
drwx-----  3 eric-chi  staff    96 Aug  7 23:15 .config/
drwxr-xr-x  15 eric-chi  staff   480 Aug  7 09:45 .dropbox/
lrwxr-xr-x   1 eric-chi  staff    55 Aug  9 22:34 .functions@ -> /Users/eric-chi/Dropbox/_igor/repos/dotfiles/functions
lrwxr-xr-x   1 eric-chi  staff    55 Aug  9 22:34 .gitconfig@ -> /Users/eric-chi/Dropbox/_igor/repos/dotfiles/gitconfig
drwxr-xr-x   5 eric-chi  staff   160 Aug  7 23:15 .npm/
-rw-----   1 eric-chi  staff    23 Aug  9 11:32 .psql_history
-rw-----   1 eric-chi  staff    7 Aug  7 22:31 .python_history
drwx-----  5 eric-chi  staff   160 Aug  8 23:44 .ssh/
drwxr-xr-x   4 eric-chi  staff   128 Aug  9 00:08 .vim/
-rw-----   1 eric-chi  staff 13344 Aug  9 22:35 .viminfo
lrwxr-xr-x   1 eric-chi  staff    51 Aug  9 22:34 .vimrc@ -> /Users/eric-chi/Dropbox/_igor/repos/dotfiles/vimrc
drwxr-xr-x   4 eric-chi  staff   128 Aug  7 21:52 .vscode/
-rw-----   1 eric-chi  staff   450 Aug  6 23:39 .zsh_history
drwx-----  3 eric-chi  staff    96 Aug  6 20:02 Applications/
drwx-----  7 eric-chi  staff   224 Aug  9 22:39 Desktop/
drwx-----+ 4 eric-chi  staff   128 Aug  8 13:04 Documents/
drwx-----+ 10 eric-chi  staff   320 Aug  9 22:06 Downloads/
drwx-----  9 eric-chi  staff   288 Aug  7 11:02 Dropbox/
drwx----- 71 eric-chi  staff  2272 Aug  7 12:16 Library/
drwx-----+ 5 eric-chi  staff   160 Aug  7 00:33 Movies/
drwx-----+ 3 eric-chi  staff    96 Aug  6 19:51 Music/
drwx-----+ 5 eric-chi  staff   160 Aug  7 17:57 Pictures/
drwxr-xr-x+  4 eric-chi  staff   128 Aug  6 19:51 Public/
drwxr-xr-x   3 eric-chi  staff    96 Aug  9 22:32 Users/
drwxr-xr-x   7 eric-chi  staff   224 Aug  9 22:34 dotfiles_old/

```

Notice the different symlinks that are created.

Awesome. We have all the symlinks created based on what we configured inside of the `make-symlinks.sh` file. Now we need to start talking about updating our files.

Step 4. Update Your Dotfiles

This part is very easy, and this is why I love this approach so much. Whenever you want to add a new update to your dotfiles, all you need to do is simply open the dotfile in your home directory like normal. Edit it as you would any other day and then save the file. Once you have added a new change into the file, it'll automatically take effect since it is a symlink, updating the file inside of your repo. So then, when you go to your dotfiles repo, you should see unstaged changes. You can then push your changes to GitHub, saving your changes and thus allowing you to always have your dotfiles backed up on GitHub and being able to pull your dotfiles onto any machine you want.

If you want to have different versions for different computers (one for work and one for personal) then this can be done as well. Just create a branch for each computer and manage it like that.

Let's say you switched computers and you want to bring all your dotfiles over. Well, it's super easy. All you need to do is install Git onto your new machine, pull down your dotfiles repo, and run the `make-symlinks.sh` file as we did in Step 3. Everything is back to normal after that's run.

You can even use this repo to store different settings. I store all my backup settings, like VSCode and iTerm. Granted, they won't work with the `make-symlinks.sh` file, but I use this repo as a one-stop shop for all my settings. I get lazy and also store larger files

like .zips on here that are relating to things like IntelliJ, using Git LFS if I have to. But overall, this is a very cool tool that allows you to version-control your dotfiles and saves you time when you are moving from computer to computer.

Well, that's that! You can make the repo private if you have sensitive information in your files or if you don't want people snooping around your dotfiles.

For the original blog, feel free to visit it here:

<http://blog.smalleycreative.com/tutorials/using-git-and-github-to-manage-your-dotfiles/>.

Source code: <https://github.com/ericjaychi/sample-dotfiles>

See you guys in the next one!

Thanks to Zack Shapiro.

Programming

Software Development

Tutorial

Github

Git

Learn more.

Medium is an open platform where 170 million readers come to find insightful and dynamic thinking. Here, expert and undiscovered voices alike dive into the heart of any topic and bring new ideas to the surface. [Learn more](#)

Make Medium yours.

Follow the writers, publications, and topics that matter to you, and you'll see them on your homepage and in your inbox. [Explore](#)

Share your thinking.

If you have a story to tell, knowledge to share, or a perspective to offer — welcome home. It's easy and free to post your thinking on any topic. [Write on Medium](#)

[About](#)

[Help](#)

[Legal](#)